Příprava elektronických výukových materiálů pro mikropočítačovou platformu Arduino

Martin Diatel

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Diatel**
Osobní číslo:	**A15715**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**prezenční**
Téma práce:	**Příprava elektronických výukových materiálů pro mikropočítačovou platformu Arduino**
Téma anglicky:	**Preparation of Electronic Educational Materials for the Arduino Microcontroller Platform**

Zásady pro vypracování:

1. **Popište existující podklady pro výuku programování mikropočítačů v oboru Bezpečnostní technologie, systémy a management.**
2. **Navrhněte rozšiřující desku pro platformu Arduino se vstupy a výstupy vyžadovanými pro výuku.**
3. **Uvedený návrh hardwarově realizujte.**
4. **Vytvořte ukázkové programové vybavení pro obsluhu použitých vstupů a výstupů.**
5. **Vypracujte podklady pro výuku s využitím vytvořené rozšiřující desky.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **BANZI, Massimo. Getting started with Arduino. 2nd ed. Farnham: O'Reilly, 2011. ISBN 9781449309879.**
2. **CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.**
3. **MARGOLIS, Michael. Arduino cookbook. 2nd ed. Sebastopol, Calif.: O'Reilly, 2012, xx, 699 p. ISBN 1449313876.**
4. **MCCONNELL, Steve. Dokonalý kód: umění programování a techniky tvorby software. Vyd. 1. Brno: Computer Press, 2005, 894 s. ISBN 802510849x.**
5. **PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300-110-1.**

Vedoucí bakalářské práce: **Ing. Jan Dolinay, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **20. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 20. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

Ing. Jan Valouch, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 27. 5. 2019 …….

podpis diplomanta

ABSTRAKT

Cílem bakalářské práce je vypracovat elektronický výukový materiál pro mikropočítačovou platformu Arduino.

Teoretická část práce se zabývá seznámením s mikropočítači, platformou Arduino a popisem současných podkladů pro výuku programování mikropočítačů v oboru Bezpečnostní technologie, systémy a management.

V praktické části byl vypracován návrh rozšiřující desky pro platformu Arduino se vstupy a výstupy vyžadovanými pro výuku, následně byl uvedený návrh hardwarově realizován. Bylo vytvořeno ukázkové programové vybavení pro obsluhu použitých vstupů a výstupů a vypracovány podklady pro výuku s využitím vytvořené rozšiřující desky.

Klíčová slova: Arduino, Mikropočítač, Rozšiřující deska, Výukový materiál,

ABSTRACT

The aim of this thesis is to develop an electronic learning material for the Arduino microcomputer platform.

The theoretical part of the thesis deals with the introduction of microcomputers, the Arduino platform and the current teaching materials for microcomputer programming in the field of Security Technologies, Systems and Management.

In the practical part, the design of the expansion board for the Arduino platform was elaborated with the inputs and outputs required for teaching, then the proposed design was implemented hardware. A sample software was developed for the use of the inputs and outputs used, and teaching materials were developed using the created extension board.

Keywords: Arduino, Microcomputers, Expansion board, Teaching materials

Touto cestou bych chtěl poděkovat vedoucímu mé bakalářské práce, panu Ing. Janu Dolinay, Ph.D. za pomoc, vstřícnost, vynaložený čas, ochotu a veškeré rady, které mi při mé práci pomohly. Dále bych rád poděkoval Zdeňkovi Havrlantovi za pomoc při výrobě rozšiřující desky. A v neposlední řadě bych chtěl poděkovat i své rodině za trpělivost, pomoc a taky za podporu, kterou mi po čas celého studia věnovali.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V současné době se téměř ve všech inteligentních elektronických zařízeních využívají mikropočítače jako jejich centrální mozek. Můžeme je najít například v telefonech, mikrovlnné troubě nebo třeba i ve světlech na jízdní kola s více možnými nastaveními pro blikání. Pro jejich programování a vytváření zdrojového kódu jsou vytvořena takzvaná vývojová prostředí. Jedním z nich je i Arduino IDE pro mikropočítačové vývojové desky stejnojmenné platformy.

Jeho výhodou je především jednoduchost a snadnost při programování, takže i začátečník nebude mít při práci žádné problémy. Navíc je zcela zdarma. Mnoho programátorů po celém světě Arduino IDE používají k vytváření vlastních programů a sdílí je na fórech, kde si je mohou ostatní stáhnout. Tím se stává Arduino pro uživatele více přívětivější.

Tato práce se zaměřuje na vytvoření výukových materiálů pro výuku předmětu Mikropočítače a PLC. Pro usnadnění práce ve výuce byla vytvořena rozšiřující deska, která obsahuje několik barevných LED diod, mikrospínače, senzor vibrací a náklonu a lineární potenciometr. Pro připojení k platformě Arduino Uno slouží oboustranné kolíky, které se jednoduše nasadí do pinů platformy. Tento způsob zapojení využívají i takzvané shieldy. Shieldy jsou rozšiřující, snadno připojitelné periférie. Obsahují další komponenty, které samotná platforma Arduino Uno neobsahuje. Rozšiřující deska byla navržena tak, aby přesně kopírovala rozměry platformy Arduino Uno a zbytečně tak nezabírala mnoho prostoru.

Dále byl vytvořen a popsán ukázkový program na obsluhu všech požadovaných vstupních a výstupních komponentů, které usnadní použití vytvořené desky ve výuce.

Pro podklady ve výuce byly vypracovány jednoduché úlohy pro studenty, ve kterých se seznámí se základními funkcemi Arduino Uno a vyzkoušejí si tak vytvoření zdrojového kódu pro ovládání jednotlivých komponentů. Všechny úlohy jsou navrženy tak, aby si studenti zkusili naprogramovat veškeré komponenty, které rozšiřující deska nabízí a zároveň aby nebyly příliš složité.

I. TEORETICKÁ ČÁST

1 MIKROPOČÍTAČE

Mikropočítač představuje zařízení, které je výrazně menší než osobní počítač a nachází se v jednom integrovaném obvodu. Obsahuje mikroprocesor, paměť, vstupní a výstupní zařízení, časovače, komunikační rozhraní apod.

V současné době se v téměř každém elektronickém zařízení využívají mikropočítače. Můžeme je najít například v počítačích, domácích kinech, autě, fotoaparátech, mobilech anebo v kalkulačkách. K jejich programování slouží mnoho různých programovacích jazyků, ale nejrozšířenější z nich jsou jazyky C nebo C++. Na osobním počítači či notebooku lze vytvořit v takzvaném integrovaném vývojovém prostředí, které obsahuje editor a nástroje pro překlad a ladění, program. [1]

V oblasti bezpečnostních technologií se mikropočítače používají téměř v každém zařízení nebo detektoru. Jsou to například pasivní infračervené detektory, kouřové hlásiče, detektory tříštění skla, infračervené závory a bariéry a samozřejmě také kamery.

1.1 Pojmy

Mikroprocesor – je procesor, který je obvykle vestavěn na jednom integrovaném obvodu. Téměř všechny současné procesory jsou tvořené jako mikroprocesory a jejich názvy jsou často zaměňovány. Jsou také známé jako CPU neboli centrální procesové jednotky. Mezi nejčastěji používané jsou mikroprocesory z řad Intel Pentium, PowerPC nebo Sun SPARC. V současné době se však rozšiřují i jiné od menších společností. [2]

Mikrokontroler – je tvořen procesorem, pamětí a vstupním/výstupním zařízením. Je součástí jednoho integrovaném obvodu určeného pro vestavění do zařízení. Jsou určené pro použití v počítačích a různých pracovních zařízeních. Nejčastěji mají paměť na vlastním čipu a poskytují pouze omezený prostor pro externí zařízení. [2]

Vstupní a výstupní periférie – vstupními perifériemi se rozumí cokoliv, co se může k systému připojit prostřednictvím kabelu nebo bezdrátově a zároveň systém ovládat nebo mu předávat informace. Může to být například klávesnice a myš od počítače, různá tlačítka nebo senzory. Výstupní periferie je vše, co systém může ovládat. Ať už se jedná o reproduktory, různá osvětlení a LED diody nebo o sirény v bezpečnostních technologiích.

Digitální vstupy a výstupy – jsou takzvané porty, které mohou být nastavovány softwarem pomocí logický nul a logických jedniček. Digitální vstupy jsou určeny ke čtení stavu vstupní

periférie: například tlačítka nebo přepínače. Výstupy jsou naopak ovládané softwarem. Nejjednodušším příkladem je změna stavu zapnuto/vypnuto. [2]

Analogové vstupy – díky analogovým vstupům je možné číst například měnící se napětí připojené periférie. U Arduina se může jednat například o potenciometr a můžeme z něj získat informace v rozmezí 0 až 5 V.

1.2 Arduino

Arduino je malá platforma od firmy Atmel, která je snadná na programování a je vhodná například pro výuku informatiky ve školách nebo i k vlastnímu použití. Pro snadné programování je navrženo vývojové prostředí IDE se sadou knihoven. Základem nejrozšířenější desky, nazvané Arduino Uno, je mikrokontroler (jednočipový mikropočítač), který obsahuje všechno potřebné pro své funkce. [3]

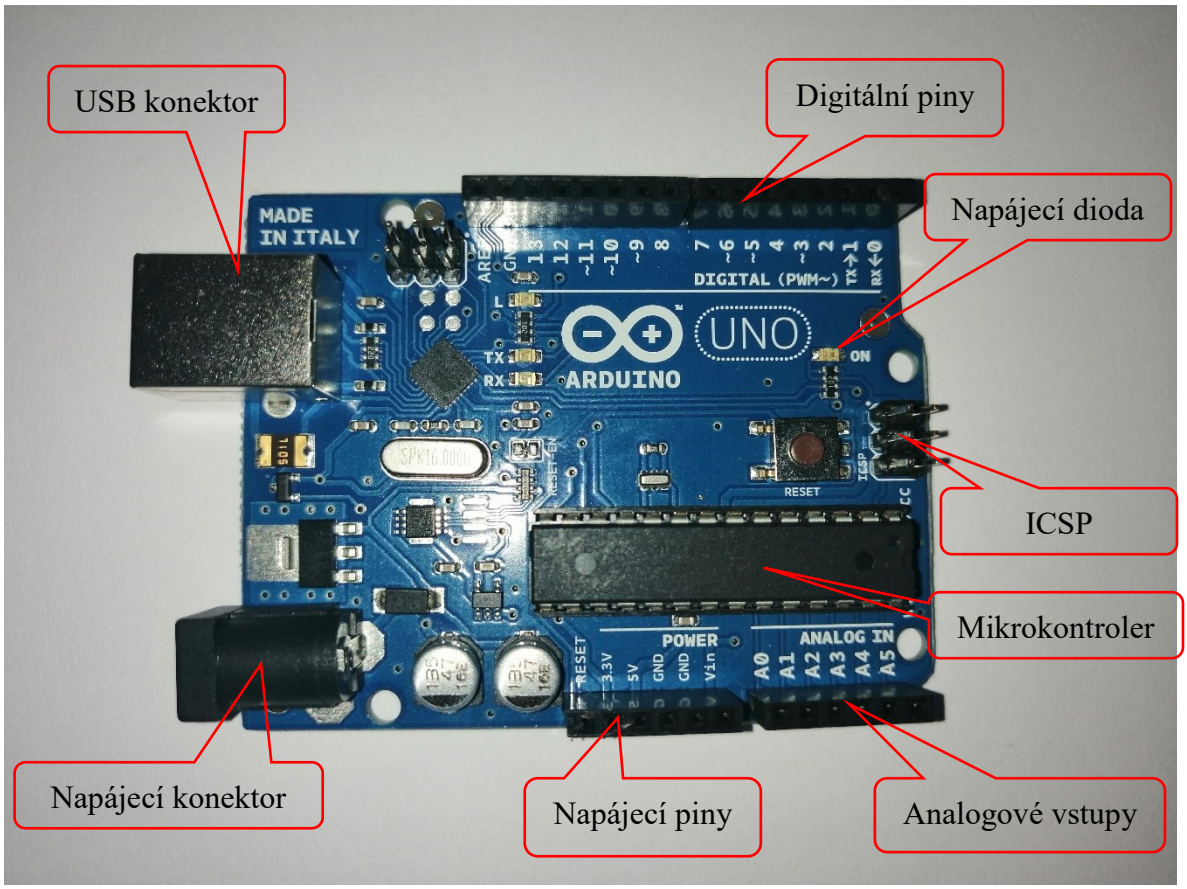
Arduino bylo navrženo tak, aby ho každý nováček bez zkušeností mohl snadno a rychle pochopit. K tomu mu mohou posloužit i ukázkové programy, které Arduino zahrnuje ve svém vývojovém prostředí. Pro pokročilejší programátory Arduino nabízí i internetovou encyklopedii, odkud si můžou nainstalovat různé knihovny pro složitější programy. Je využíváno k programování mnoho různých zařízení jako jsou světla, hry, inteligentní zařízení nebo roboti. Jako rozdíl od ostatních vývojových desek, Arduino využívá ke komunikaci místo sériového portu USB port. Výhodou je, že spousta uživatelů vlastní vývojovou desku Arduino a je proto velmi jednoduché najít mnoho ukázkových programů, takže jeho použití je opravdu snadné. [4]

Existuje více verzí platforem Arduino, které se liší především funkcemi a cenou. Navíc lze každou platformu doplnit o další komponenty, čímž se zvýší možnosti jejich využití. Jedním z komponentů je například senzor vibrací a náklonu SW-520D, který je použit ve vytvořené rozšiřující desce. Dále lze použít například potenciometr, který se dá mimo jiné použít k ovládání intenzity osvětlení pomocí změny napájení.

1.2.1 Arduino Uno

Arduino Uno je první ze série mikrokontrolérových desek USB Arduino založených na ARmega328. Jeho název „Uno“ pochází z italského jazyka ve kterém znamená první a byl vybrán, aby označil číslo vydání. Deska obsahuje 14 digitálních vstupních / výstupních pinů, 6 analogových vstupů, 16MHz krystal, připojení pomocí USB, napájecí konektor, ICSP rozhraní, resetovací tlačítko, diodu indikující napájení a diodu na pinu 13, kterou lze ovládat z

programu. Jeho vstupní napětí je maximálně 6 až 20 V a pracovní 5 V. Paměť je omezena na 32 KB, ale pro programování v předmětu bezpečnostní technologie, systémy a management je to dostatečný prostor. [5]



Obr. 1: Platforma Arduino s popisem součástek

2 SOUČASNÉ PODKLADY PRO VÝUKU PROGRAMOVÁNÍ MIKROPOČÍTAČŮ V OBORU BTSM

V současných podkladech pro výuku programování mikropočítačů v oboru Bezpečnostní technologie, systémy a management se můžeme seznámit mikropočítačovou platformou Arduino a vyzkoušet si ji. Součástí podkladů jsou návody a rozsáhlé vypracované úlohy. Máme možnost se zde seznámit s vybranými funkcemi Arduina a naučit se je používat v praxi.

2.1 Popis předmětu

Cílem předmětu Mikropočítače a PLC je získat základní znalosti z oblasti mikropočítačových systémů a programovatelných automatů v oblasti řízení technologických procesů a seznámit studenty s možnostmi použití mikropočítačů a PLC automatů v oblasti monitorování a řízení technologických procesů.

Mezi základní cíle předmětu patří pochopení základních pojmů z mikroprocesorové techniky, a to zejména logické funkce.

Ve výuce se studenti učí programovat ve vývojovém prostředí předpřipravené úkoly. Prvním nejjednodušším úkolem je rozblikání LED diody. Druhým rozsáhlejším úkolem je vytvořit poplachový systém ke střežení vzácných předmětů. Systém lze stisknutím tlačítka nebo příkazu přes sériovou linku zajistit nebo odjistit. Při vyvolaní poplachu se musí zadat předem dané heslo k jeho zrušení. Pomocí potenciometru, kterým se nastavuje čas, se určuje doba, za kterou se má systém sám zajistit. Závěrečným úkolem je vypracovat projekt, ve kterém se ověřují získané znalosti z programování. [6]

Studenti mají k dispozici vývojovou desku Arduino Uno a sadu součástek, která se skládá ze tří LED diod různých barev, dvou tlačítek, potenciometru, několika rezistorů, spojovacích drátků a nepájivého pole, do kterého se umísťují součástky.

2.2 Rozšiřující deska

Hlavní nevýhodou současných podkladů v oboru Bezpečnostní technologie je nutnost manuálně připojit rozšiřující desku pomocí přívodních kabelů a zapojit do ní jednotlivé součástky. Při zapojení mohou vznikat chyby, které by mohly některé z komponentů nebo dokonce i samotnou platformu Arduino Uno znehodnotit.

Vytvořením rozšiřující desky se tak minimalizuje čas a zabrání se výskytu možných chyb při zapojování. Bakalářská práce se kvůli možnému vzniku nebo výskytu chyb, zabývá

vývojem rozšiřující desky, která může tyto chyby odstranit nebo je alespoň z větší části minimalizovat. Práce ve výuce se tak mnohonásobně zrychlí a studenti se budou moci naučit mnohem více věcí.

Oproti současným výukovým materiálům rozšiřující deska obsahuje předem připravené vstupní a výstupní periférie a odpadá tedy nutnost cokoliv zapojovat. Jediné, co musí uživatel udělat je, že ji jednoduše nasune do platformy Arduino Uno a začne programovat. Rozšiřující deska má stejný rozměr jako Arduino Uno a tudíž vůbec neomezuje prostor. Obsahuje také senzor vibrací a náklonu, který v současných materiálech není.

II. PRAKTICKÁ ČÁST

3 NÁVRH ROZŠIŘUJÍCÍ DESKY PRO PLATFORMU ARDUINO

Pro zjednodušení práce na výuce byla vytvořena rozšiřující deska s požadovanými vstupními a výstupními komponenty.

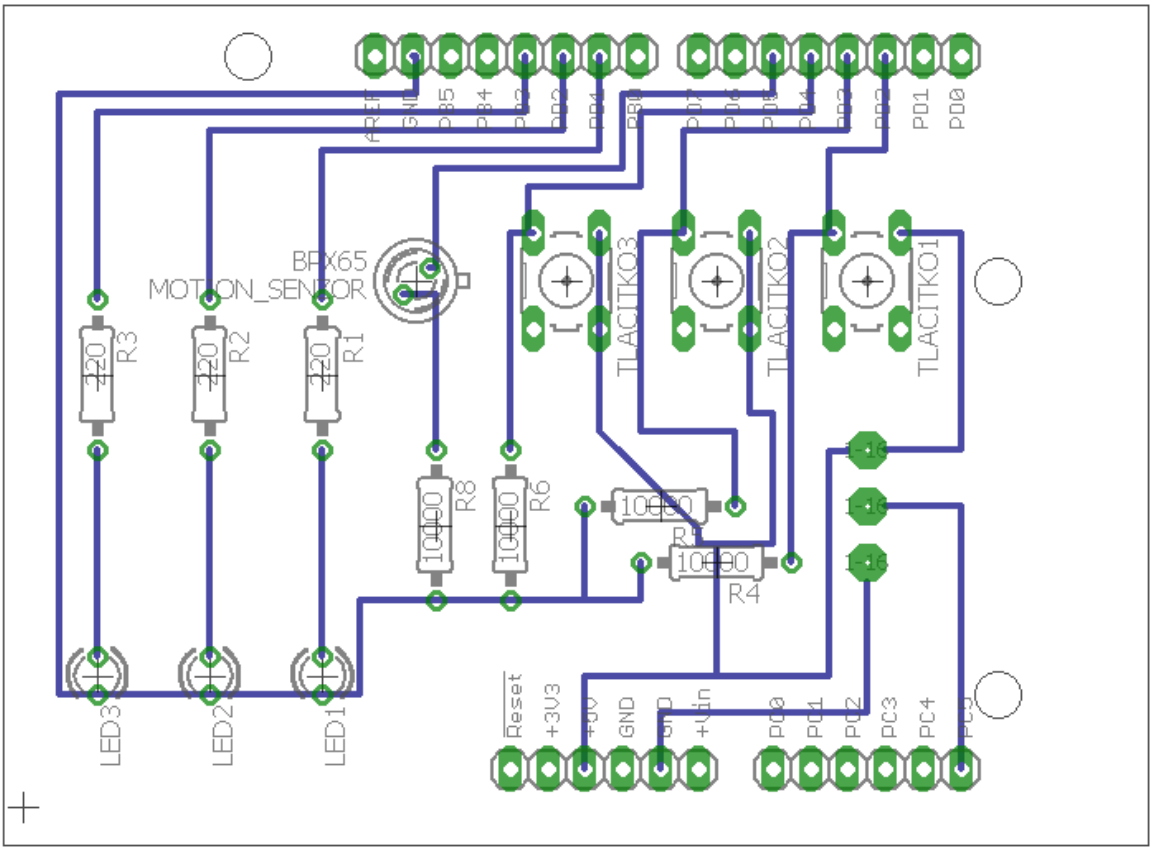
Použité komponenty:

- dvoubarevné difuzní LED diody o velikosti 3 mm a napájením 5 V,
- mikrospínače SMD jedno pólové s polohami ON-OFF,
- rezistory pro LED diody SMA 0207 o velikosti odporu 220 Ω,
- rezistory pro mikrospínače SMA 0207 o velikosti odporu 10 000 Ω,
- senzor vibrací a náklonu SW-250D reagující na vibrace nebo náklon,
- lineární potenciometr o velikosti odporu 10 000 Ω,
- oboustranné kolíky S1G40S s roztečí 2,54 mm.

Jednotlivé komponenty jsou popsány na obrázku číslo 4.

3.1 Schéma desky plošných spojů

Schéma desky plošných spojů bylo vytvořeno v editoru plošných spojů EAGLE. Komponenty byly rozmístěny tak, aby byla zaručena snadná obsluha při programování.



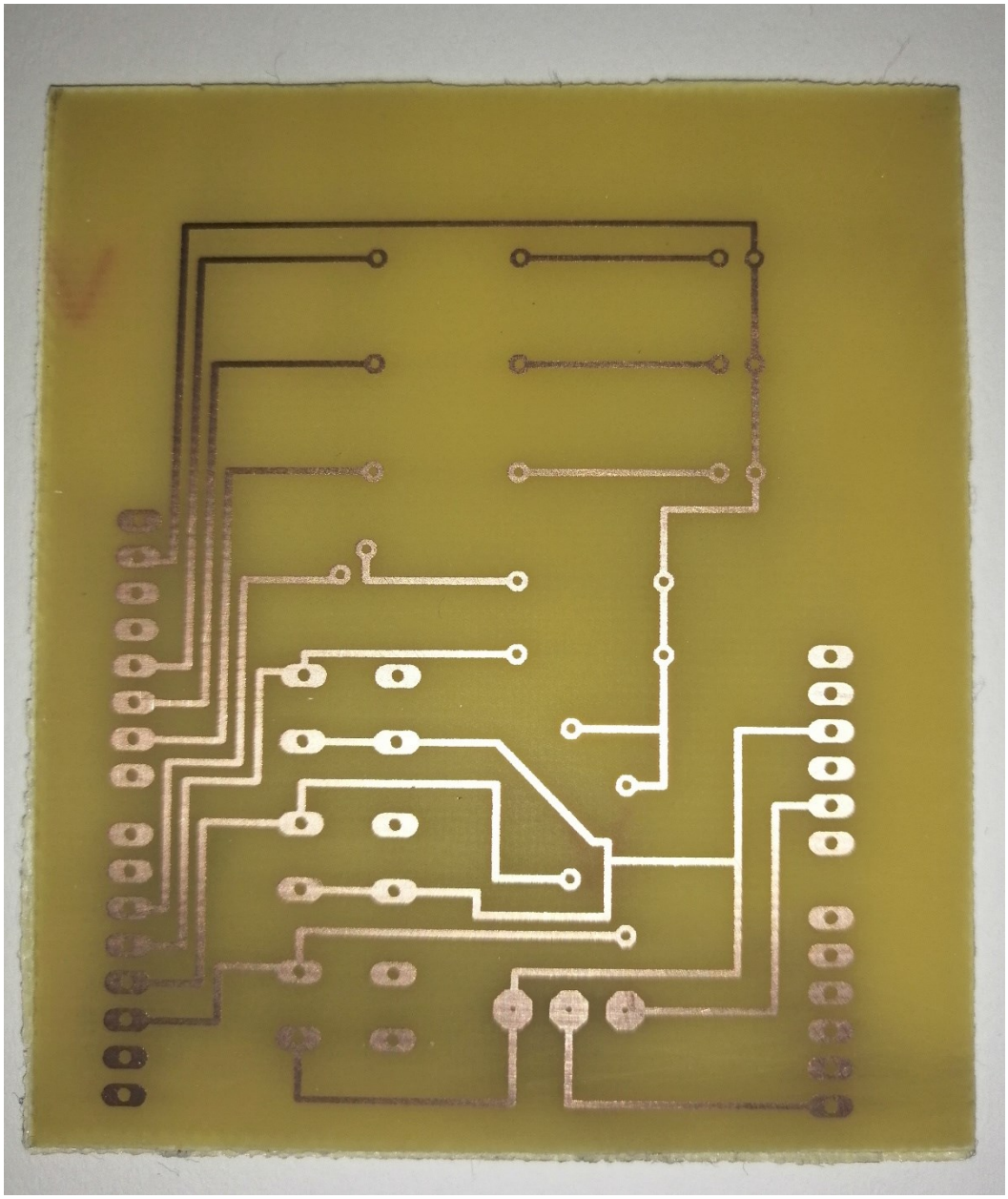
Obr. 2: Návrh desky plošných spojů

Tabulka 1: Prvky s příslušnými piny

Název	Příslušný pin
TLACITKO1	2
TLACITKO2	3
TLACITKO3	4
SENZOR	5
LED1	9
LED2	10
LED3	11
POTENCIOMETR	A5

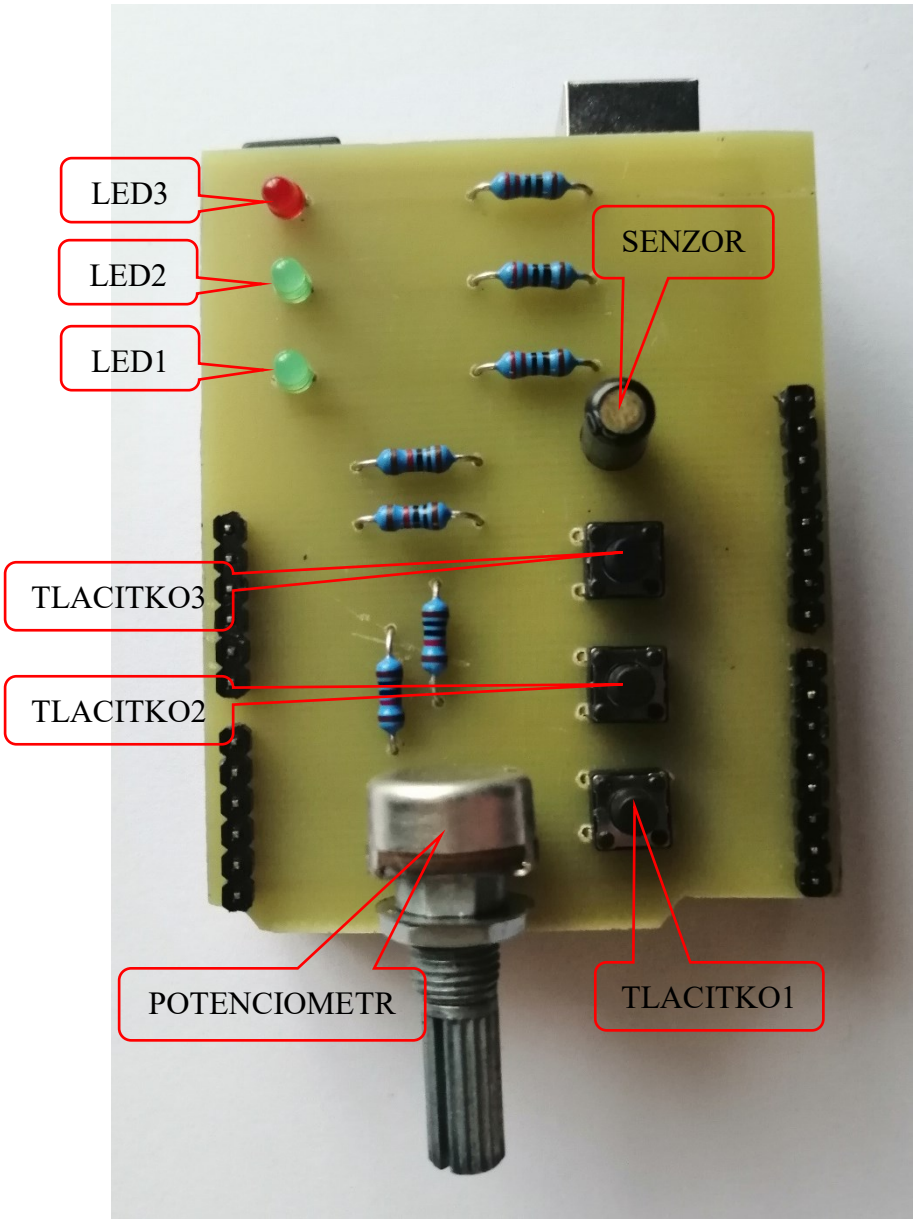
4 HARDWAROVÁ REALIZACE ROZŠIŘUJÍCÍ DESKY

Na základní cuprexitovou desku bylo vytištěno propojení jednotlivých komponentů tak, aby se při následném leptání v roztoku odstranila pouze veškerá přebytečná měď a komponenty zůstaly propojené. Dalším krokem bylo vyvrtání děr pro komponenty a jejich osazení do desky. Posledním závěrečným krokem bylo připájení všech komponentů a otestování funkčnosti.

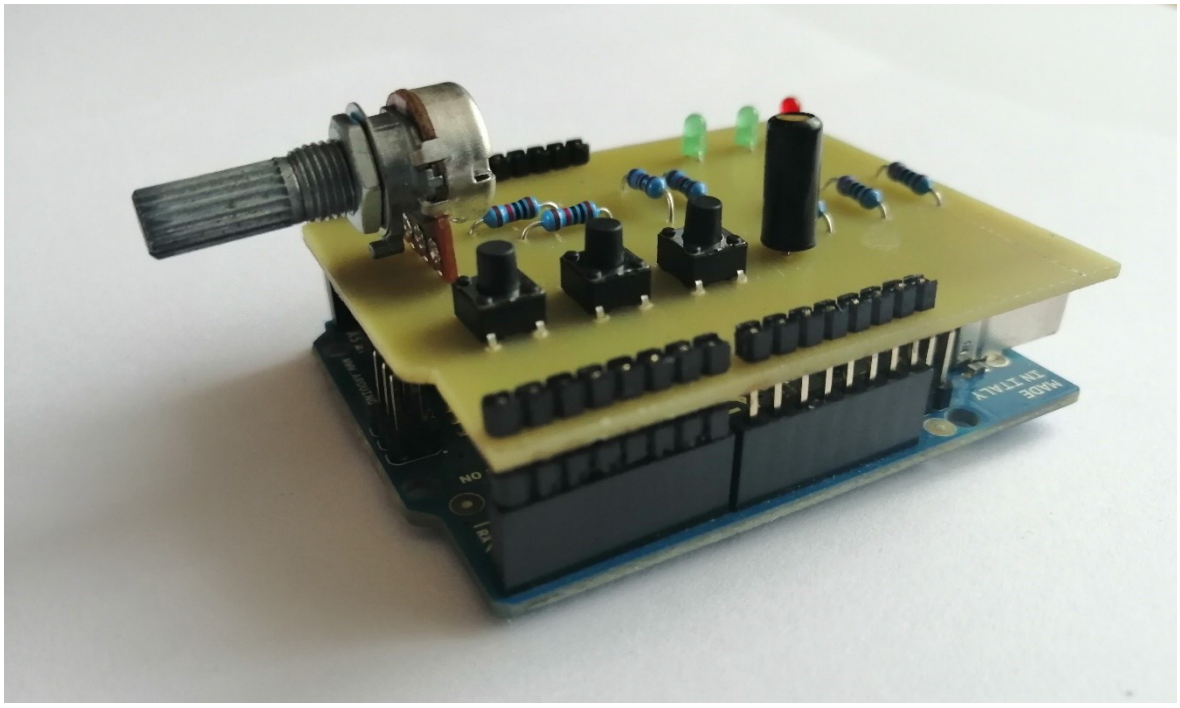


Obr. 3: Cuprexitová deska s propojením jednotlivých komponentů

Deska, jak už bylo zmíněno, obsahuje tři barevné LED diody, které se dají ovládat například pomocí tří tlačítek, které jsou na desce rovněž umístěny. Dále se na zde nachází senzor vibrací a náklonu, jenž může sloužit jako poplachový senzor při neoprávněné manipulaci. Poslední částí je lineární potenciometr, kterým se může například měnit jas diody nebo nastavit doba, za kterou se má provést nějaká akce.



Obr. 4: Hotová rozšiřující deska



Obr. 5: Hotová rozšiřující deska s Arduino Uno

5 UKÁZKOVÉ PROGRAMOVÉ VYBAVENÍ PRO OBSLUHU

Pro ukázku všech funkcí rozšiřující desky bylo vypracováno ukázkové programové vybavení.

V mé bakalářské práci je deska použita jako zabezpečovací systém s tlakovým senzorem na střežení cenného objektu. Je vybavena kontaktem proti sabotáži v případě, že by se pachatel rozhodl odstranit kryt senzoru. Pokud by pachatel manipuloval s celým systémem, vyhlásí se poplach díky senzoru vibrací a náklonu. Systém lze tlačítkem odjistit nebo zajistit. Potenciometrem se dá nastavit doba, za kterou se má vyhlásit poplach při zvednutí předmětu z tlakového senzoru. Ve všech případech poplachu se rozbliká červená LED dioda. První zelená LED dioda indikuje svitem napájení a druhá stav systému (zajištěno/odjištěno). Do sériového monitoru se následně vypisují všechny stavy, konkrétněji to jsou:

- [SYSTEM] Zařízení bylo spuštěno.
- [SYSTEM] Zařízení bylo zajištěno.
- [SYSTEM] Zařízení bylo odjištěno.
- [ALARM] Pohybový senzor byl aktivován.
- [ALARM] Tamper byl aktivován.
- [ALARM] Tlakový senzor byl aktivován.
- [INFO] Čas pro zrušení poplachu: počet sekund s.
- [ALARM] Vyvolaní poplachu.
- [SYSTEM] Zrušení poplachu.

Důvodem zvolení ukázkového programu byla demonstrace funkcí, jenž se podobají funkcím, které mají detektory v oblasti bezpečnostních technologiích.

5.1 Ukázkový program

Při připojení napájení se rozsvítí LED dioda, která je napojena na pinu 9 a zároveň se do sériového monitoru vypíše zpráva o spuštění. Pomocí tlačítka na pinu 2 se systém uvádí do stavů zajištěno nebo odjištěno a rovněž se do sériového monitoru vypíše zpráva v jakém stavu se systém nachází. Pokud je systém zajištěn, svítí LED dioda napojená na pinu 10. Tlačítko na pinu 3 slouží jako takzvaný „Tamper“ kontakt a detekuje sabotáž při pokusu odstranění krytu systému. V případě zjištění jeho aktivace se rozbliká červená LED dioda napojená na pinu 11, čímž indikuje poplach a do sériového monitoru se vypíše zpráva o jeho aktivaci. Poslední tlačítko na pinu 4 je tlakový senzor. Pokud je tlačítko uvolněno, systém

spustí čas, za který se má vyhlásit poplach a v sériovém monitoru se zobrazí zpráva o aktivaci tlakového senzoru a doba, za kterou se spustí alarm. Tato doba se nastavuje pomocí potenciometru, který je napojen na analogovém vstupu A5. Po uplynutí této doby se rozbliká červená dioda a zobrazí se zpráva o vyvolání poplachu. Během nastaveného času lze poplach zrušit. Senzor vibrací a náklonu napojený na pinu 5 detekuje jakoukoliv manipulaci se systémem. Jakmile se senzor aktivuje, systém vyhlásí poplach a opět se zobrazí zpráva o stavu v sériovém monitoru.

```
//Definování vstupních a vystupních pinů názvy. Slouží pro snadnější orientaci v kódu
#define LED_ALARM             11
#define LED_SECURED           10
#define LED_POWER             9
#define BUTTON_TAMPER         3
#define BUTTON_SECURED        2
#define BUTTON_PRESSURE       4
#define SENSOR_MOTION         5
#define TIMER                 A5
//Definování typů stavů pro snadnější programování
#define STATE_NORMAL          0
#define STATE_ALARM_TAMPER    1
#define STATE_ALARM_PRESSURE  2
#define STATE_ALARM_MOTION    3
//Vtvoření proměn
bool isSecured = false;//Určuje zda je systém zajišten nebo odjištěn [true = zajištěn, false = odjištěn]
bool isAlarm = false;//Určuje zda je vyvolán poplach [true = je poplach, false = není poplach]
bool getStateSensor[4];//Vytvoření seznamu, který určuje, zda byl na sériovou linku vypsána hláška
bool buttonSecuredPressed = false;//Slouží pro kontrolu, zda bylo staděno tačítko pro zajištění
bool isTimer = false;//Slouží pro zahájení "příchodového spoždnění"
bool ledAlarm = false;//Určuje, zda se má led alarmu (červená) rozsvítit či zhasnout
int stateAlarm = 0, oldStateAlarm = 0;//Slouží pro zjištění současného stavu systému
int timeSecured;//Hodnota je čas pro příchodové zpoždění
long int getTime;//Slouží pro uložení času příchodového zpoždění
long int timeBlink;//Slouží pro uložení času pro rozsvícení nebo zhasnutí LED diody alarmu
```

Obr. 6: Ukázkový program 1. část

```
void setup() {
  Serial.begin(9600);//Nastavení rychlosti komunikace sériové linky
  //Určení funkce pinů, zda li se jedná o vstup nebo výstup
  //Nastavení výstupů
  pinMode(LED_ALARM, OUTPUT);
  pinMode(LED_SECURED, OUTPUT);
  pinMode(LED_POWER, OUTPUT);
  //Nastavení vstupů
  pinMode(BUTTON_TAMPER, INPUT);
  pinMode(BUTTON_SECURED, INPUT);
  pinMode(BUTTON_PRESSURE, INPUT);
  pinMode(SENSOR_MOTION, INPUT_PULLUP);
  pinMode(TIMER, INPUT);

  digitalWrite(LED_POWER, HIGH);//Rozsvítí LED pro indikaci, že je zařízení spuštěno

  Serial.println("[SYSTEM] Zařízení bylo spuštěno");//Vypíše zprávu na seriovou linku
}
//Vytvoření funkce která vrací při stisknutí tlačítka "true" a při klidové poloze "false"
bool pressButton(int idButton) {
  if (digitalRead(idButton) == HIGH) {//Vytvoření podmínky, zda je na daném pinu napětí
    return true;//Pokud napětí na pinu je, vrací hodnotu "true"
  } else {
    return false;//Pokud napětí na pinu není vrací hodnotu "false"
  }
}
```

Obr. 7: Ukázkový program 2. část

```
//Funkce pro vypsaní poplachové hlášky na sériovou linku a pro ovládání části systému
void changeStatus() {
  switch (stateAlarm) {
    case STATE_NORMAL://Pokud přejde systém do normalního režimu
      digitalWrite(LED_ALARM, LOW);//Nastaví LED diodu alarmu do vypnutého stavu
      isAlarm = false;
      break;
    case STATE_ALARM_TAMPER://Pokud přejde system do poplachu kvůli rozpojení tamper kontaktu
      Serial.println("[ALARM] Tamper byl aktivován");
      isAlarm = true;
      break;
    case STATE_ALARM_PRESSURE://Pokud přejde system do poplachu kvůli aktivaci tlakového senzoru
      Serial.println("[ALARM] Tlakový senzor byl aktivován");
      Serial.print("[INFO] Čas pro zrušení poplachu:");
      Serial.println(String(timeSecured) + "s");
      isTimer = true;//Spuštení odpočtu pro příchodové zpoždění
      getTime = millis();//Nastaví do proměnné čas od spuštění Arduina v milisekundách
      break;
    case STATE_ALARM_MOTION://Pokud přejde systém do poplachu kvůli aktivaci pohybového senzoru
      Serial.println("[ALARM] Pohybový senzor byl aktivován");
      isAlarm = true;
      break;
  }
}
```

Obr. 8: Ukázkový program 3. část

```
void loop() {
  /*Nastavení do proměnné příchodový čas
     Maximální hodnota analogového vstupu je 1024 a minimální 0,
     proto pro zjištění procentuální hodnoty vydělíme hodnotu analogového vstupu 1024.
     Poté hdnotu vynasobíme 10 a zaokrouhlíme na realné číslo.
     Při otáčení potenciometrem získame hodnoty od 0-9, proto příčteme +1 aby byla minimální hodnota 1 a maximalní 10
  */
  timeSecured = int(float(analogRead(TIMER)) / 1024 * 10) + 1;

  if (isTimer) {//Pokud je "isTimer" nastaven na "true"
    /*Pokud je součet času od spuštění odpočtu a nastavené hodnoty potenciometru menší jak
       aktualní čas od spuštení Arduina.
    */
    if (getTime + (timeSecured * 1000) <= millis()) {
      isAlarm = true;//Spustí poplach
      isTimer = false;//Zastaví se odpočet příchodového zpoždění
      Serial.println("[ALARM] Vyvolaní poplachu");
    }
  }

  if (isSecured) {//Zjištění zda je system zajištěn či ne
    digitalWrite(LED_SECURED, HIGH);//Zdali je system zajišten rozsvítí se LED dioda (zelená)
  } else {
    digitalWrite(LED_SECURED, LOW);//Zdali je system odjištěn zhasne se LED dioda (zelená)
    stateAlarm = 0;//Nastaví systém do normálního režimu, čímž zruší polach
  }
```

Obr. 9: Ukázkový program 4. část

```
if (isAlarm) {//Zjištění zda je vyvolán poplach
  //Podmínka pro rozblíkání LED diody při poplachu
  if (timeBlink + 500 <= millis()) {
    timeBlink =  millis();
    ledAlarm = ! ledAlarm;
  }
  if (ledAlarm) digitalWrite(LED_ALARM, HIGH); else digitalWrite(LED_ALARM, LOW);
}

if (pressButton(BUTTON_SECURED)) {//Funkce pro ověření stisknutí tlačitka pro zajištění nebo odjištění systému
  /*
     Podmínka pro zamezení opakování kódu pokud je tlačítko trvale stlačeno
  */
  if (!buttonSecuredPressed) {
    buttonSecuredPressed = true;
    isSecured = ! isSecured;
    if (!isSecured) {
      if (isAlarm) Serial.println("[SYSTEM] Zrušení poplachu");
      Serial.println("[SYSTEM] Zařízení bylo odjištěno");
    } else {
      Serial.println("[SYSTEM] Zařízení bylo zajistěno");
    }
  }
} else {
  buttonSecuredPressed = false;
}
```

Obr. 10: Ukázkový program 5. část

```
if (!pressButton(BUTTON_TAMPER) and isSecured) {//Podmínka pro zjištění, zda je aktivován tamper
  if (!getStateSensor[STATE_ALARM_TAMPER]) {//Zjištění, zda už byl jednou poplach vyhlášen
    stateAlarm = STATE_ALARM_TAMPER;
    getStateSensor[STATE_ALARM_TAMPER] = true;
  }
} else {
  if (!isSecured and getStateSensor[STATE_ALARM_TAMPER]) getStateSensor[STATE_ALARM_TAMPER] = false;
}
if (!pressButton(BUTTON_PRESSURE) and isSecured) {//Podmínka pro zjištění, zda je tlakový senzor aktivován
  if (!getStateSensor[STATE_ALARM_PRESSURE]) {//Zjištění, zda už byl jednou poplach vyhlášen
    stateAlarm = STATE_ALARM_PRESSURE;
    getStateSensor[STATE_ALARM_PRESSURE] = true;
  }
} else {
  if (!isSecured and getStateSensor[STATE_ALARM_PRESSURE]) getStateSensor[STATE_ALARM_PRESSURE] = false;
}

if (digitalRead(SENSOR_MOTION) == HIGH  and isSecured) { //Zjištění, zda byl aktivován pohybový senzor
  if (!getStateSensor[STATE_ALARM_MOTION]) {//Zjištění, zda už byl jednou poplach vyhlášen
    stateAlarm = STATE_ALARM_MOTION;
    getStateSensor[STATE_ALARM_MOTION] = true;
  }
} else {
  if (!isSecured and getStateSensor[STATE_ALARM_MOTION]) getStateSensor[STATE_ALARM_MOTION] = false;
}
```

Obr. 11: Ukázkový program 6. část

```
  if (stateAlarm != oldStateAlarm) {//Podmínka pro zjištení, zda došlo k novému poplachu
    oldStateAlarm = stateAlarm;
    changeStatus();
  }
}
```

Obr. 12: Ukázkový program 7. část

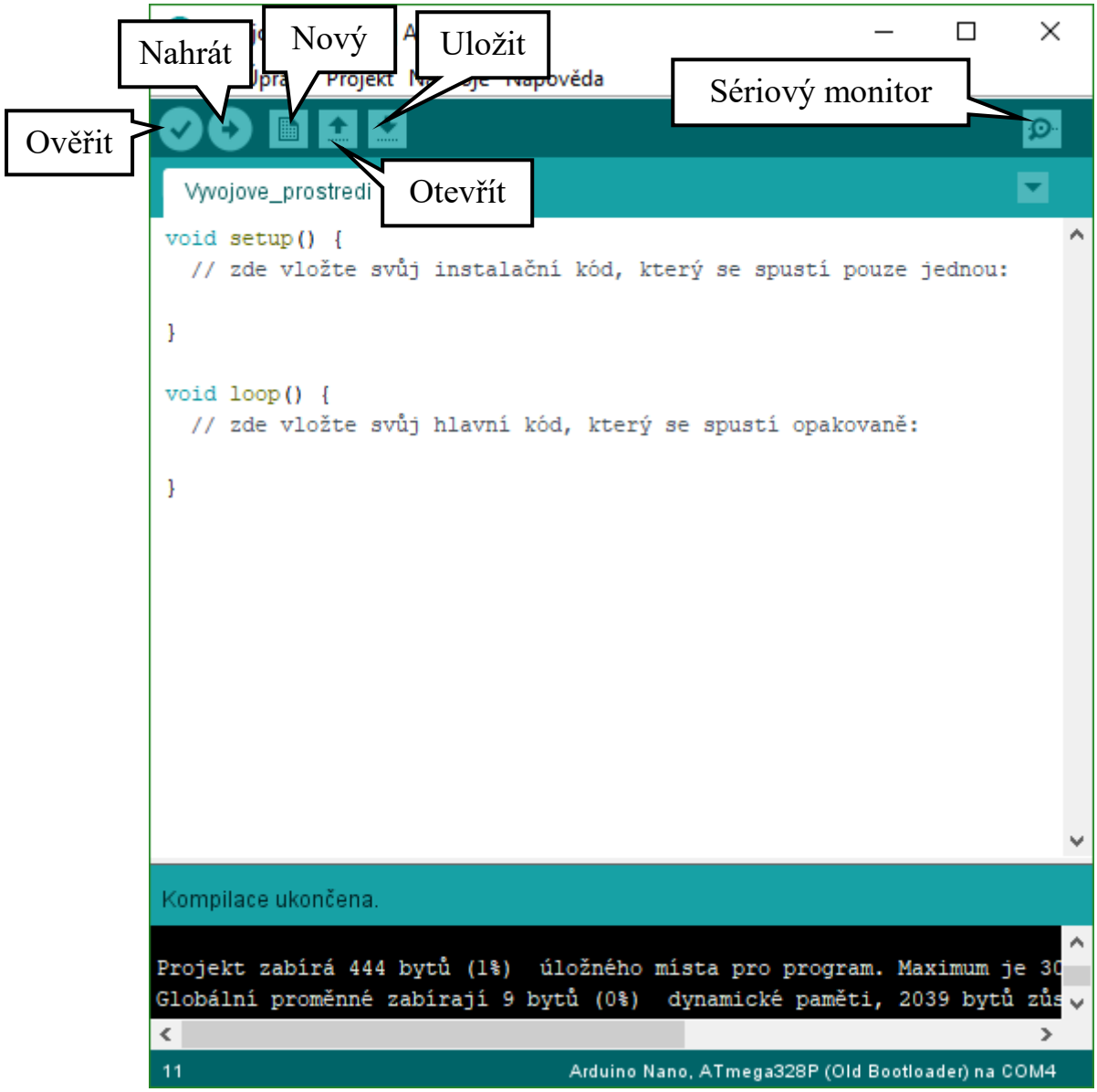
6 PODKLADY PRO VÝUKU S VYUŽITÍM VYTVOŘENÉ ROZŠIŘUJÍCÍ DESKY

Následující kapitola popisuje vývojové prostředí, definice a ukázky základních funkcí, které jsou nutné pro fungování vytvořené rozšiřující desky. To může následně představovat i možné podklady pro výuku a tím i usnadnit práci ve výuce.

6.1 Vývojové prostředí

Vzhledem k tomu, že každá mikropočítačová platforma vyžaduje vývojové prostředí pro programování a určité znalosti, je následující část práce zaměřena na popsání rozhraní vývojového prostředí a charakteristiku základní struktury.

Samotné vývojové prostředí je program, který je spuštěn v počítači a s jehož pomocí vytváříme zdrojový kód programu. Ve světě Arduino se tento vývojový program nazývá sketch neboli náčrtek. Vytvořené zdrojové kódy se poté nahrají do platformy Arduino.



Obr. 13: Vývojové prostředí

Rozhraní:

- **Ověřit –** tlačítko ověřit nám slouží k takzvanému překladu programu. Stisknutím tohoto tlačítka se spustí kompilace a převede se nám zapsaný kód do strojového kódu, který Arduino využívá pro svou práci. Zároveň si tak můžeme ověřit, zda máme kód správně napsán. V opačném případě se v dolní části vývojového prostředí zobrazí chybová hláška s popisem problému.

- **Nahrát –** toto tlačítko slouží k ověření a přenosu napsaného kódu do připojené desky (Arduino Uno). Tlačítko ověřit tedy můžeme vynechat a rovnou kliknout na tlačítko nahrát, to však bude fungovat pouze je-li připojena deska.
- **Nový –** spustí nové okno vývojového prostředí pro zápis nového kódu.
- **Otevřít –** tímto tlačítkem otevřeme menu, ve kterém si můžeme vybrat, zda chceme otevřít dřívější program, na kterém jsme pracovali, anebo spustit předem připravené ukázkové programy.
- **Uložit –** slouží k okamžitému uložení současného programu.
- **Sériový monitor –** pokud uživatel potřebuje znát aktuální stav nebo jakoukoliv hodnotu, kterou si nechal vypsat pomocí programu, může si ho jednoduše zobrazit do sériového monitoru pomocí příkazů.

Struktury:

Základní struktury vývojového prostředí, které se automaticky vytvoří při spuštění nového programu, jsou setup() a loop(). Je důležité, aby se kód vkládal mezi závorky { a }, které značí začátek a konec určité funkce.

- **setup()** – tato funkce se provádí po startu programu a to pouze jednou. Spustí se po nahrání programu, resetu a po odpojení a znovu připojení napájení. Používá se k inicializaci proměnných, režimů pinů, spuštění knihoven apod.
- **loop()** – funkce se provádí opakovaně od začátku až po konec programu. Jakmile dojde na poslední příkaz, opakuje se znovu. [7]

Struktury řízení:

- **if/else –** příkaz if se používá k vytvoření podmínky a provádění příkazu, pokud je podmínka pravdivá. Může se používat zároveň s příkazem else, který se provede, pokud je první podmínka if nepravdivá. Lze také kombinovat více příkazů else pod sebou.

```
if (podmínka) {
    vykonej tvrzení A;
} else {
    vykonej tvrzení B;
}
```

Obr. 14: Správná syntaxe if/else

```
void loop() {
  if (tlacitko == HIGH) {           //Pokud je tlačítko zmáčknuto
     digitalWrite(LED,HIGH);        //Rozviť diodu
  } else {
     Serial.println("LED nesviti"); //Pokud není zmáčknuto napiš LED nesvítí
  }
}
```

Obr. 15: Příklad zápisu příkazu if/else

- **for –** příkaz for slouží k opakování části programu v uzavřených závorkách. Může se používat zároveň s počítadlem přírůstků a následné ukončení smyčky.

```
for (inicializace; stav; přírůstek) {
  // tvrzení
}
```

Obr. 16: Správná syntaxe zápisu

```
for(int i = 0; i > 5; i++) {                // Smyčka se bude opakovat dokud nebude
                                            // splněna podmínka i > 5 a při každém
                                            // průchodu se hodnota i zvedne o 1
  Serial.println("Promenna i je mensi nez 5");//Při každém průchodu bude
                                            //vypisovat do sériového monitoru
}
  Serial.println("Podminka je splnena");     //Po splnění podmínky vypíše
                                            //do sériového monitoru Podminka je splnena
```

Obr. 17: Příklad zápisu příkazu for

- **break –** break se používá k výstupu ze smyček for, while a do. [7]

Syntaxe:

- **#define** – je užitečná direktiva C ++, která umožňuje programátorovi zadat název konstantní hodnotě před kompilací programu. Definované konstanty v arduinu nezabírají žádné místo v paměti programu na čipu. Kompilátor nahradí odkazy na tyto konstanty definovanou hodnotou v době kompilace.

```
#define trvalýNazev       hodnota
#define tlacitko         (5)
```

Obr. 18: Syntaxe a příklad použití #define

- **#include –** se používá k zahrnutí externích knihoven do programu a dává uživateli možnost přístupu ke knihovnám vytvořených přímo pro Arduino a ke standartním

knihovnám z jazyku C. Tyto knihovny se dají volně nainstalovat z oficiálních stránek a fór.

```
#include <NázevKnihovny>
#include <WiFi.h>       //Přístup do knihovny pro využití WiFi
```

Obr. 19: Syntaxe a příklad použití #include

- **;** – tento symbol nám při programování slouží k ukončení příkazu. Je velmi důležitou součástí a bez jeho zadání by nám vyskočilo chybové hlášení.
- **{ a }** – složené závorky jsou důležitou částí při programovaní. Je nutné, aby každá začáteční závorka měla i svou koncovou závorku. Při hledání chyby s chybějící závorkou stačí pouze najet kurzorem myši na počáteční závorku a koncová závorka se automaticky zvýrazní. Používá se například u funkcích jako jsou if, for, else, do a další.
- **// a /* */** – slouží programátorovi ke psaní poznámek přímo ve vývojovém prostředí. Dvou lomítek za sebou se používá, pokud chceme vytvořit komentář pouze na jednom řádku. Druhé možnosti se využívá, pokud máme delší komentář. Komentář bude začínat za symboly „/*“ a bude pokračovat, dokud ho zase neukončíme dvojicí symbolů „*/“. [7]

```
//Komentář, který nám stačí pouze na jeden řádek
/*Komentář který pokračuje
 * dokud ho opět neukončíme
 * na jekémkoliv řádku*/
```

Obr. 20: Příklad psaní komentářů

Datové typy:

- **int** – neboli **integer,** je celočíselný datový typ a umožňuje zapsat do proměnné celé číslo o velikosti 16 bitů a umí ukládat i záporná čísla v rozsahu od -32 768 do 32 767.

```
int NázevProměnné = hodnota;
int CeleCislo     = 5234;
```

Obr. 21: Syntaxe a příklad deklarace int

- **long –** je datový typ do kterého můžeme zapsat delší celé číslo než do datového typu int. Jeho délka může být v rozsahu od -2 147 483 648 až do 2 147 483 647. Jeho velikost je 32 bitů.

```
long NázevProměnné   = hodnota;
long DlouheCeleCislo= 523456;
```

Obr. 22: Syntaxe a příklad deklarace long

- **float –** je datový typ, který slouží k zápisu čísla s desetinou čárkou. Maximální počet všech číslic je 6 až 7 a celková velikost činí 32 bitů. Doporučuje se číslům s desetinnou čárkou vyhnout, jelikož práce s nimi zabere programu více času než s celými čísly. Desetinné číslo se při programování rozděluje tečkou.

```
float NázevProměnné    = hodnota;
float DesetinneCislo   = 0.12345;
```

Obr. 23: Syntaxe a příklad deklarace float

- **bool –** je datový typ, který má dvě hodnoty true a false (pravda a nepravda). Každá proměnná má velikost 8 bitů.
- **char –** používá se, pokud potřebujeme do programu uložit nějaké písmeno nebo řetězec znaků. Znaky jsou v Arduinu IDE převedeny na čísla jako v ASCII tabulce a můžeme s nimi tak i pracovat. [7]

```
char NázevProměnné   = 'Znak';
char PismenoA        = 'A';
char PisemnoA        = 65; //hodnota písmena A v ASCII tabulce
```

Obr. 24: Syntaxe a příklad deklarace char

Práce s časem:

- **delay() –** funkce slouží k zastavení programu na zadanou dobu a udává se v milisekundách. Příklad zápisu: „delay(1000);“. Tento zápis nám říká, aby se program zastavil na 1000 milisekund neboli na 1 sekundu a poté pokračoval dál. Může být použit například pro rozblikání diody.
- **mikros(), millis() –** tyto funkce se používají ke zjištění času od zapnutí desky. Jak už z názvu vyplívá, první funkce vrátí počet mikrosekund a druhá počet milisekund od zapnutí. [7]

Funkce:

- **pinMode** – pomocí této funkce můžeme nastavit určitý pin (vývod mikropočítače) aby se choval jako vstup nebo výstup. Pokud použijeme pin jako vstup, můžeme k němu připojit například tlačítko a zjistit tak, zda je stisknuté anebo ne. Druhou možností je využití pinu jako výstup a připojit k němu například diodu a následně tak ovládat, zda svítí nebo ne.
    - Důležité je zachovat správnou syntaxi zápisu. V tomto případě „pinMode(pin, mode);“, kde „pin“ je číslo pinu, který chceme ovládat a „mode“ jeho režim. Nejčastější režimy této funkce jsou INTPUT a OUTPUT (vstup a výstup).

```
pinMode(číslo pinu,vstup/výstup);
pinMode(9,OUTPUT);  //pin číslo 9 je výstupní
```

Obr. 25: Syntaxe a příklad deklarace pinMode

- **digitalWrite** – zapisuje na určitém pinu hodnotu HIGH nebo LOW. To znamená, že se na pin přivede napětí 5 V (neboli logická 1) a může rozsvítit diodu nebo 0 V (neboli logická 0). Jelikož se jedná o digitální pin, můžeme na něj zapsat pouze jeden ze dvou možných stavů.
    - Správná syntaxe zápisu je: „digitalWrite(pin, value);“, kde za „pin“ dosadíme číslo pinu, který chceme ovládat a za „value“ hodnotu, kterou na něj chceme poslat.

```
digitalWrite(číslo pinu,stav);
digitalWrite(9,HIGH); //pošle na pin číslo 9 napětí 5V
```

Obr. 26: Syntaxe a příklad deklarace digitalWrite

- **digitalRead** – příkaz čte hodnotu určitého vstupního pinu. Jelikož se opět jedná o digitální pin, příkaz nám vrátí hodnoty HIGH a LOW. Tímto způsobem můžeme zjistit například zda je stisknuto tlačítko. [7]

```
digitalRead(číslo pinu);
digitalRead(9); //přečte hodnotu pinu číslo 9
```

Obr. 27: Syntaxe a příklad deklarace digitalRead

Komunikace

- **Serial** – slouží pro sériovou komunikaci s monitorem ve vývojovém prostředí Arduina. Důležitou částí je rychlost přenosu dat v bitech, která nám v našem případě bude stačit 9600 bitů za sekundu a zadává se příkazem „Serial.begin();“. Pro vypsání textu na sériový monitor se pak používá příkaz „Serial.println(text);“. [7]

```
Serial.begin(9600);       //Nastaví rychlost přenosu dat
Serial.println("Arduino") //Zobrazí text Arduino v sériovém monitoru
```

Obr. 28: Příklad deklarace Serial.begin a Serial.println

7 UKÁZKOVÉ ÚLOHY ZE ZÁKLADŮ PROGRAMOVÁNÍ

Níže jsou vytvořeny ukázkové úlohy pro studenty, ve kterých se mohou seznámit se základy programování a mohou si také vyzkoušet všechny prvky, které rozšiřující deska nabízí.

7.1 Úloha č. 1: Program na ovládání LED diod z vývojového rozhraní

Zadání: Vytvořte program, pomocí kterého rozsvítíte diodu na pinu 9 pomocí příkazu ve vývojovém prostředí.

Postup:

Použijeme funkce pinMode a digitalWrite. Do funkce setup mezi závorky { a } napište tento příkaz:

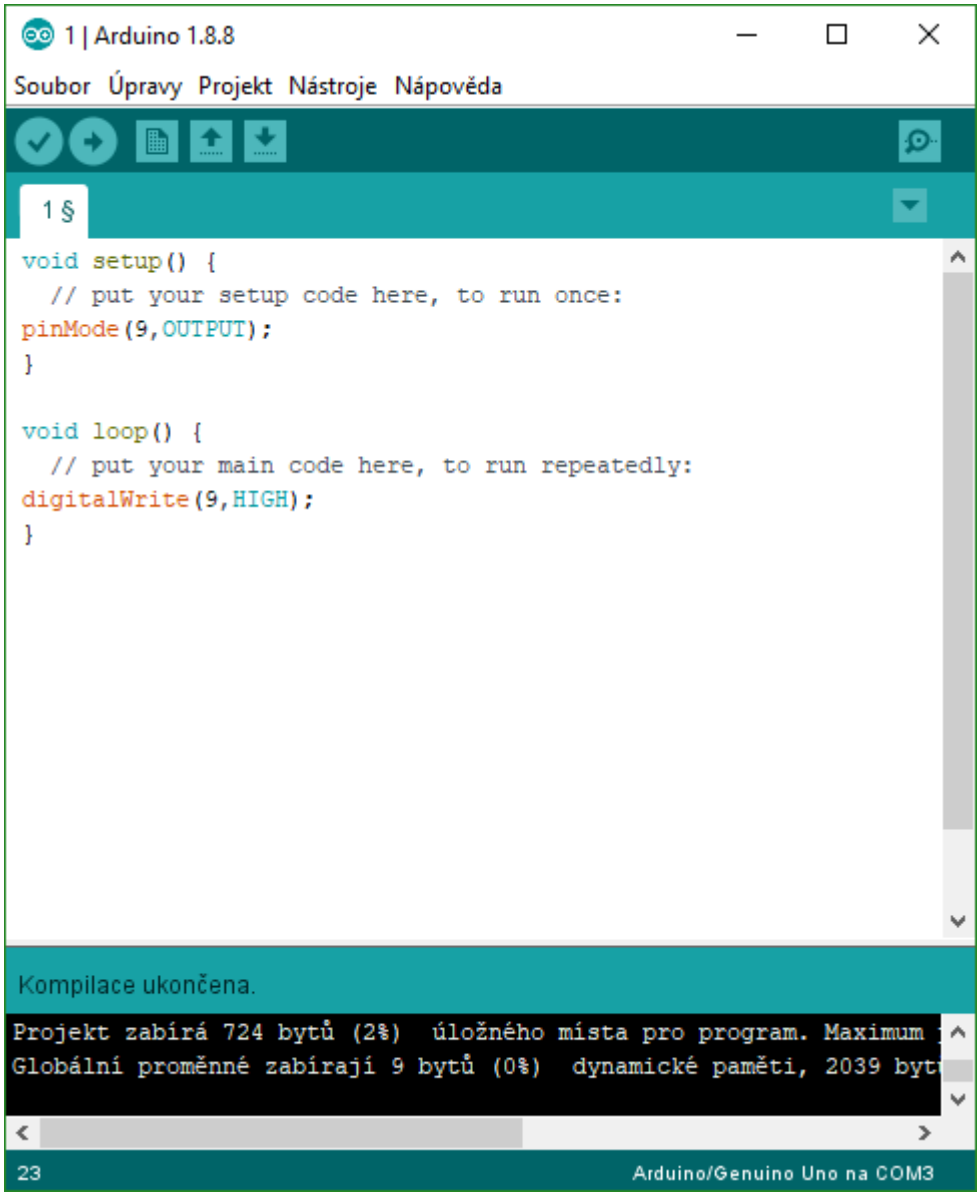
```
pinMode(9,OUTPUT);
```

Tento příkaz nám nastaví pin číslo 9 jako výstup. Dbejte na to, aby byla zachována velká a malá písmena.

Do funkce loop mezi závorky { a } zadejte následující příkaz:

```
digitalWrite(9,HIGH);
```

Tímto příkazem odešleme na pin 9 logickou jedničku. Následně program uložte pomocí tlačítka **Uložit** a pomocí tlačítka **Ověřit** ověřte program. Pokud se ve spodní části vývojového prostředí nezobrazí chyba, program máte správně zapsán.



Obr. 29: Ukázka prvního programu

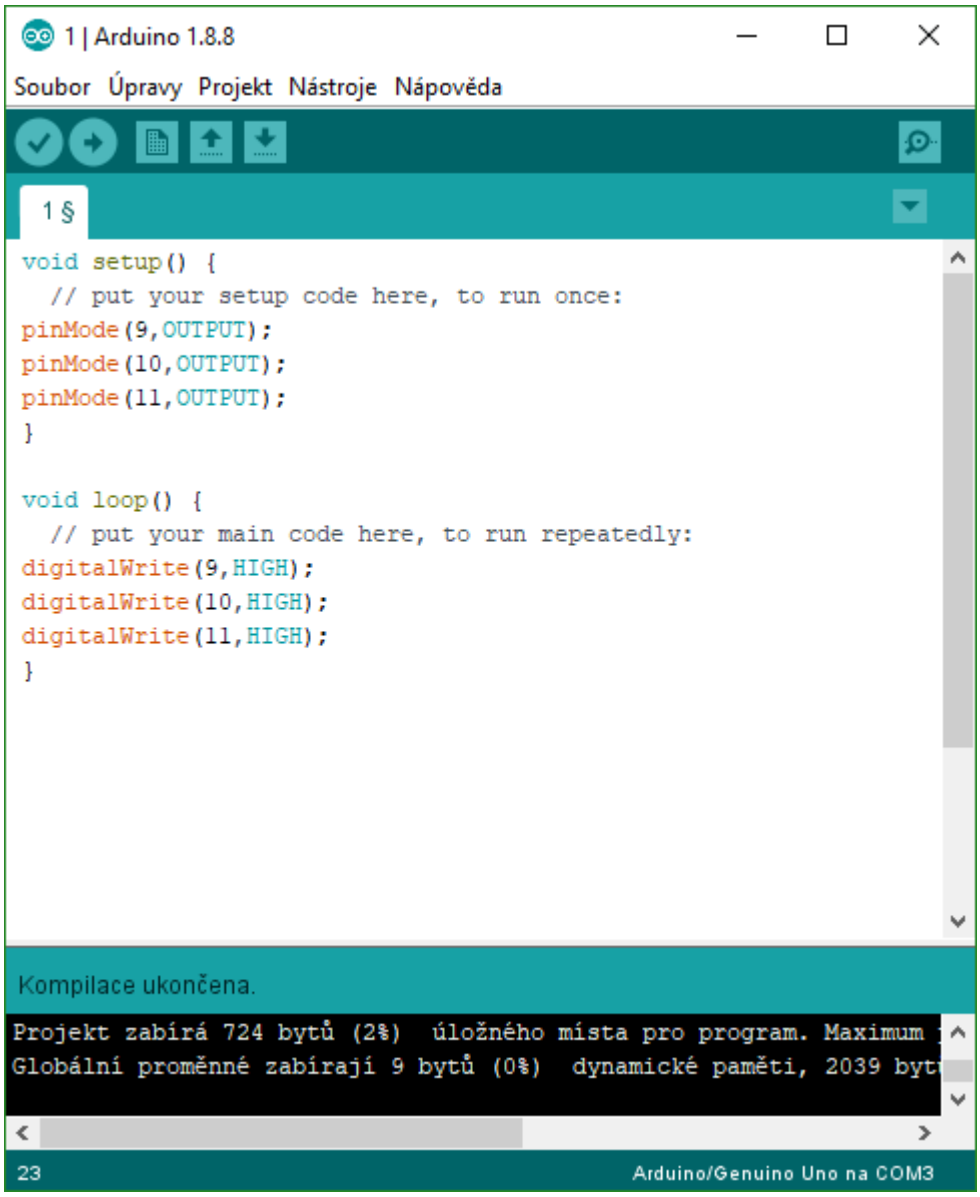
Jakmile si program ověříte, připojte Arduino pomocí USB kabelu k počítači a pomocí tlačítka **Nahrát** program nahrajte. Po nahrání programu by měla červená LED dioda začít svítit. Pokud se tak nestalo, dioda může být vadná nebo špatně zapsaný program.

Zadání II: Vytvořte program, pomocí kterého rozsvítíte všechny diody na pinech 9, 10 a 11 pomocí příkazu ve vývojovém prostředí.

Postup:

Postup je totožný s předešlým postupem, jen se přidají další dvě diody. Nastavíme všechny piny, na kterých jsou umístěny diody jako výstupní. Následně nastavíme jejich hodnotu na

HIGH. Nakonec program ověřte a nahrajte do Arduina. Pokud je správně zapsán, rozsvítí se všechny tři diody.



Obr. 30: Ukázka druhého programu

Zadání III: Vytvořte program, pomocí kterého rozblikáte všechny diody na pinech 9, 10 a 11 pomocí příkazu ve vývojovém prostředí.

Postup:

K vytvoření programu použijeme předešlý program, který doplníme o funkci delay(500) pro práci s časem. V této funkci zvolíme požadovaný čas na 500 milisekund, což nám pro rozblikání diod stačí. Pokud bychom však nezměnili hodnotu diod na logickou nulu, program

by pouze diody postupně rozsvítil. Proto po funkci delay změníme jejich hodnotu na LOW. Funkci delay(500) vložíme do funkce loop následujícím způsobem:

```
void loop() {
        digitalWrite(9,HIGH);
        delay(500);
        digitalWrite(9,LOW);
        digitalWrite(10,HIGH);
        delay(500);
        digitalWrite(10,LOW);
        digitalWrite(11,HIGH);
        delay(500);
        digitalWrite(11,LOW);
}
```

Jakmile máme program zapsán, opět ověříme a nahrajeme do Arduina. Diody začnou postupně blikat podle toho, jak jsme je pod sebe zapsali. Tedy nejprve dioda na pinu 9, poté na pinu 10 a naposled na pinu 11.

7.2 Úloha č. 2: Program na ovládání LED diod s využitím tlačítek

Zadání: Vytvořte program, pomocí kterého budete ovládat diody na pinech 9, 10 a 11 pomocí tlačítek na rozšiřující desce, které jsou na pinech 2, 3 a 4. Tlačítko na pinu 2 bude rozsvěcovat diodu na pinu 9 a postupně následující tlačítka i další následující diody.

Postup:

Pro zjednodušení práce si všechny komponenty definujeme pod jedinečným názvem pomocí direktivy #define. Tato funkce se zadává na úplný začátek programu nebo kdekoliv v jeho části. Avšak pro snadnou práci při delších programech je lepší si příslušné komponenty definovat na začátku.

```
#define LED1 (9)
#define LED2 (10)
#define LED3 (11)
#define TLACITKO1 (2)
#define TLACITKO2 (3)
#define TLACITKO3 (4)
```

Pro zjištění, zda je tlačítko zmáčknuto, použijeme funkci digitalRead. Nejprve si však musíme definovat tlačítka jako vstupy pomocí pinMode.

```
pinMode(TLACITKO1,INPUT);
```

Tímto příkazem jsme nastavili tlačítko na pinu 2 jako vstupní, tudíž po zmáčknutí tlačítka vyšleme do systému logickou jedničku. Ale abychom mohli zjistit, zda je tlačítko opravdu zmáčknuté, musíme použít další funkci digitalRead. Tato funkce nám vrací hodnotu ze vstupu, tedy logickou jedničku nebo nulu.

```
digitalRead(TLACITKO1);
```

Pro provedení však potřebujeme vytvořit i podmínku, při které se zjistí, zda je tlačítko stisknuto a podle toho rozsvítí diodu. To provedeme pomocí podmínky if a else následujícím způsobem:

```
void loop() {
        if(digitalRead(TLACITKO1) == HIGH) {
                digitalWrite(LED1,HIGH);
         }else{
        digitalWrite(LED1,LOW);
        }
}
```

Tato podmínka nám zaručí, že pokud je tlačítko na pinu 2 zmáčknuté, rozsvítí se dioda na pinu 9. Pokud tlačítko není zmáčknuté, dioda svítit nebude. Obdobně pak můžeme vytvořit podmínku pro všechny tlačítka.

```
void setup() {
  // put your setup code here, to run once:
pinMode(TLACITKO1,INPUT);
pinMode(TLACITKO2,INPUT);
pinMode(TLACITKO3,INPUT);
pinMode(LED1,OUTPUT);
pinMode(LED2,OUTPUT);
pinMode(LED3,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(digitalRead(TLACITKO1) == HIGH){
    digitalWrite(LED1, HIGH);
  }else{
  digitalWrite(LED1, LOW);
  }
    if(digitalRead(TLACITKO2) == HIGH){
    digitalWrite(LED2, HIGH);
  }else{
  digitalWrite(LED2, LOW);
  }
    if(digitalRead(TLACITKO3) == HIGH){
    digitalWrite(LED3, HIGH);
  }else{
  digitalWrite(LED3, LOW);
  }
}
```

Obr. 31: Ukázka třetího programu

7.3 Úloha č. 3: Program na ovládání LED diod pomocí tlačítek a vypsání stavu na sériový monitor

Zadání: Vytvořte program, pomocí kterého budete ovládat diody na pinech 9, 10 a 11 pomocí tlačítek na rozšiřující desce, které jsou na pinech 2, 3 a 4. Tlačítko na pinu 2 bude rozsvěcovat diodu na pinu 9 a postupně následující tlačítka i další následující diody. Zároveň se každé zmáčknutí tlačítka vypíše na sériový monitor.

Postup:

Pro vytvoření programu využijeme již předchozí vypracovaný program. Pro komunikaci se sériovým monitorem slouží příkazy Serial.begin pro spuštění komunikace a definování rychlosti přenosu dat a Serial.println pro vypsání textu do něj.

Jako první si vytvoříme proměnné, do kterých si uložíme stav, zda je konkrétní tlačítko stisknuto nebo ne. Jako proměnná nám poslouží datový typ bool, který má dvě hodnoty true nebo false. Pro každé tlačítko vytvoříme jednu proměnnou.

```
bool tlacitko1 = false;
bool tlacitko2 = false;
bool tlacitko3 = false;
```

Tuto hodnotu chceme změnit v případě, že je tlačítko stisknuto. Proto ji změníme v podmínce, kde rozsvěcujeme diodu.

```
 if (digitalRead(TLACITKO1) == HIGH) {
   digitalWrite(LED1, HIGH);
   tlacitko1 = true;
```

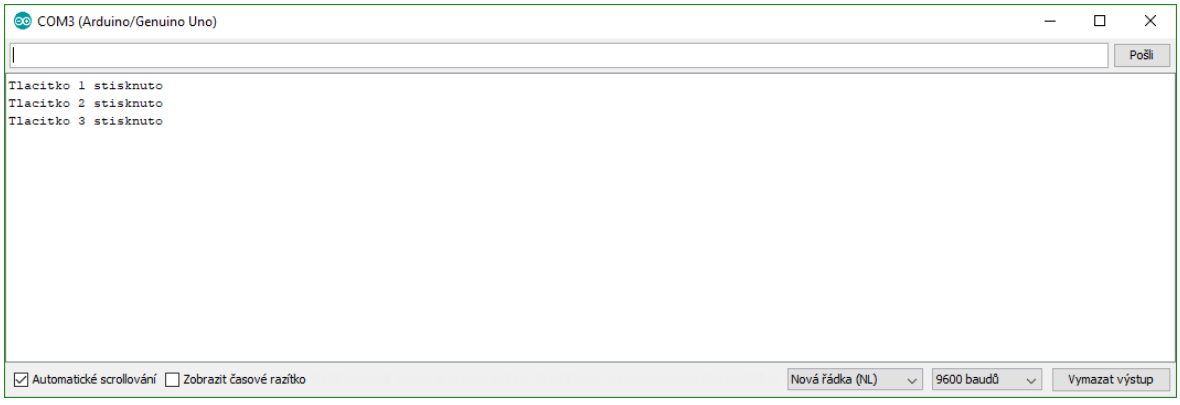
Dalším krokem je vypsání textu do sériového monitoru, které se spustí pouze jednou, pokud je tlačítko stisknuto. K tomu si vytvoříme novou podmínku. Za touto podmínkou pak ale musíme vrátit tlačítko zpět na false, jinak by se nám neustále vypisoval text do sériového monitoru.

```
   if (tlacitko1) Serial.println("Tlacitko 1 stisknuto");
   tlacitko1 = false;
```

To samé vytvoříme i u dalších dvou tlačítek. Samotný kód bude nakonec vypadat následovně:

```
bool tlacitko1 = false;
bool tlacitko2 = false;
bool tlacitko3 = false;
void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(TLACITKO1) == HIGH) {
    digitalWrite(LED1, HIGH);
    tlacitko1 = true;
  } else {
    digitalWrite(LED1, LOW);
    if (tlacitko1) Serial.println("Tlacitko 1 stisknuto");
    tlacitko1 = false;
  }
  if (digitalRead(TLACITKO2) == HIGH) {
    digitalWrite(LED2, HIGH);
    tlacitko2 = true;
  } else {
    digitalWrite(LED2, LOW);
    if (tlacitko2) Serial.println("Tlacitko 2 stisknuto");
    tlacitko2 = false;
  }
  if (digitalRead(TLACITKO3) == HIGH) {
    digitalWrite(LED3, HIGH);
    tlacitko3 = true;
  } else {
    digitalWrite(LED3, LOW);
    if (tlacitko3) Serial.println("Tlacitko 3 stisknuto");
    tlacitko3 = false;
  }
}
```

Obr. 32: Program s výpisem na sériový monitor



Obr. 33: Výpis do sériového monitoru

7.4 Úloha č. 4: Program na změnu jasu LED diody pomocí potenciometru

Zadání: Vytvořte program, pomocí kterého budete moci měnit jas diody na pinu 11 potenciometrem, který je na pinu A5.

Postup:

Definujeme si LED diodu a potenciometr.

```
#define LED3 (11)
#define POTENCIOMETR (A5)
  pinMode(LED3, OUTPUT);
  pinMode(POTENCIOMETR, INPUT);
```

Potenciometr využívá analogový vstup, jelikož jeho otáčením měníme napětí, které propouští. Tudíž budeme využívat příkazy analogRead a analogWrite. Jelikož analogRead u potenciometr vrací hodnoty od 0 do 1024 a analogWrite může zapisovat pouze hodnoty do 255, musíme hodnotu z potenciometru nejprve vydělit číslem 1024 a následně ji vynásobit 255. Vydělením hodnoty číslem 1024 však můžeme získat i desetinné číslo. Proto si ji převedeme do datového typu float. Po vynásobení ji ale převedeme do datového typu int, aby se číslo zaokrouhlilo. Nakonec tuto hodnotu pomocí analogRead nastavíme na LED diodu.

```
#define LED3 (11)
#define POTENCIOMETR (A5)

void setup() {
  // put your setup code here, to run once:
  pinMode(LED3, OUTPUT);
  pinMode(POTENCIOMETR, INPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(LED3, int((float(analogRead(POTENCIOMETR)) / 1024) * 255));
}
```

Obr. 34: Změna jasu diody potenciometrem

7.5 Úloha č.5: Program na zjištění pohybu

Zadání: Vytvořte program, který bude zaznamenávat změnu při pohybu s rozšiřující deskou a v případě zjištění pohybu vypište do sériového monitoru text „Byl zaznamenán pohyb“. Zároveň se také rozsvítí všechny LED diody. Senzor vibrací a náklonu je připojen na pinu 5.

Postup:

Senzor vibrací a náklonu se v podstatě chová jako tlačítko. Při zjištění pohybu se změní hodnota z 0 na 1. Proto s ním můžeme pracovat jako s tlačítkem. Nejprve si definujeme senzor vibrací a náklonu a nastavíme ho jako vstupní periférii.

```
#define SENZOR_POHYBU (5)
  pinMode(SENZOR_POHYBU, INPUT);
```

Poté opět jako v minulé úloze vytvoříme proměnnou bool do které se nám bude ukládat stav.

```
bool pohyb = false;
```

Následně vytvoříme podmínku, díky které zjistíme, zda byl zaznamenán pohyb. V podmínce rovněž nastavíme, aby se rozsvítily všechny diody a do sériového monitoru byl vypsán požadovaný text.

```
#define LED1 (9)
#define LED2 (10)
#define LED3 (11)
#define SENZOR_POHYBU (5)

void setup() {
  // put your setup code here, to run once:
  pinMode(SENZOR_POHYBU, INPUT);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  Serial.begin(9600);
}
bool pohyb = false;
void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(SENZOR_POHYBU) == HIGH) {
    digitalWrite(LED1, HIGH);
    digitalWrite(LED2, HIGH);
    digitalWrite(LED3, HIGH);
    pohyb = true;
  } else {
    digitalWrite(LED1, LOW);
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    if (pohyb) Serial.println("Byl zaznamenan pohyb");
    pohyb = false;
  }
}
```

Obr. 35: Zjištění pohybu

ZÁVĚR

Cílem bakalářské práce bylo vytvoření výukových materiálů pro mikropočítačovou platformu Arduino vhodné pro použití ve výuce předmětu Mikropočítače a PLC.

Teoretická část se zabývá základními informacemi ohledně mikropočítačů a jejich využití v praxi v bezpečnostních technologiích. Byly popsány jejich charakteristické vlastnosti a funkce. Hlavním cílem bakalářské práce bylo konkrétně představit platformou Arduino, která byla následně využita k vytvoření výukových materiálů, a to z důvodu dostupnosti a jednoduchosti práce v rozhraní Arduino. V další části byl popsán současný stav podkladů pro výuku v předmětu Mikropočítače a PLC.

Praktická část bakalářské práce popisuje vytvoření rozšiřující desky pro platformu Arduino Uno, jejíž funkcí je minimalizace chyb při zapojování a také zkrácení času nutného pro zapojení. Postup při její výrobě byl následující: nakreslení schéma plošných spojů a následné vytištění na cuprexitovou desku. Po vytištění byly do desky pomocí vrtačky vyvrtány díry. Do vytvořených děr se vložily požadované vstupní a výstupní komponenty. Konkrétně se jednalo o difuzní diody, mikrospínače, rezistory pro LED diody a mikrospínače, senzor vibrací a náklonu, lineární potenciometr a oboustranné kolíky, a to pro snadné propojení s platformou Arduino Uno. Nakonec byly všechny komponenty připájeny a byla otestována jejich funkčnost.

Jako demonstrace všech možných funkcí rozšiřující desky byl vypracován ukázkový program, ve kterém jsou využity všechny požadované vstupní a výstupní komponenty.

Následně byly také vypracovány podklady pro výuku založenou na nově vytvořené rozšiřující desce. V těchto podkladech pro výuku bylo nejprve popsáno vývojové prostředí, ve kterém se vytváří zdrojový kód. Dále zde byly popsány základní funkce a direktiva, které se v programování Arduino používají. Jako poslední část byly vytvořeny ukázkové úlohy pro studenty, ve kterých se seznámí se základy programování a vyzkouší si všechny prvky, které rozšiřující deska nabízí.

SEZNAM POUŽITÉ LITERATURY

[1] Cvičení 1. In: *Návod, cv. 1* [online]. Zlín: Dolinay, 2017, s. 10 [cit. 2019-05-15]. Dostupné z: http://vyuka.fai.utb.cz/pluginfile.php?file=%2F49330%2Fmod_resource%2Fcontent%2F4%2Fnavod_cviceni_1.pdf

[2] CATSOULIS, John. Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.

[3] Arduino. *Robot klub Rychnov* [online]. Rychnov: Locker, 2011, 14. 12. 2013 [cit. 2019-05-25]. Dostupné z: http://robotika.vosrk.cz/guide/arduino/cs.

[4] MARGOLIS, Michael. Arduino cookbook. 2nd ed. Sebastopol, Calif.: O'Reilly, 2012, xx, 699 p. ISBN 1449313876.

[5] ARDUINO UNO REV3. *Arduino* [[online]. Italy: Banzi, 2019, 2019 [cit. 2019-05-15]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

[6] *Výuka na FAI UTB ve Zlíně* [online]. Zlín, 2019 [cit. 2019-05-25]. Dostupné z: https://vyuka.fai.utb.cz

[7] Language Reference. Arduino [online]. Italy, c2019 [cit. 2019-05-23]. Dostupné z: https://www.arduino.cc/reference/en/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BTSM Bezpečnostní technologie, systémy a management.

CPU Central Processing Unit.

IDE Integrated Development Environment

ICSP In-Circuit Serial Programming

LED Light-Emitting Diode

PLC Programmable Logic Controller

SMD Surface Mount Device

USB Universal Serial Bus

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH